Cluster Techniques

OpenACC

More Science, Less Programming

Last updated: September 22 2022

Cluster Techniques

Contents

What is a Cluster?

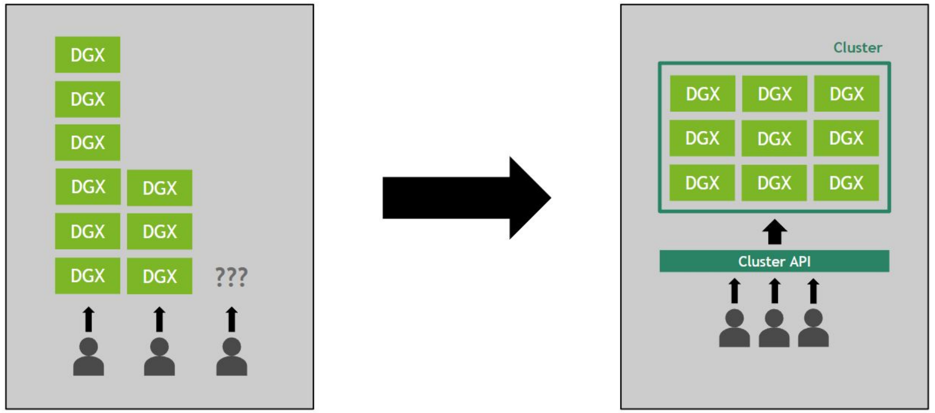
A computer cluster is a set of connected computers, usually connected by a high-speed network, that work together as if they are a single, much more powerful machine. A computer cluster may range from a simple two-node system connecting two personal computers to a supercomputer.

Why use a Cluster?

A computer cluster can provide numerous advantages over a dedicated single server, such as:

- Faster processing speeds
- Wider availability of resources
- Greater reliability

Why use a Cluster?



Dedicated Nodes

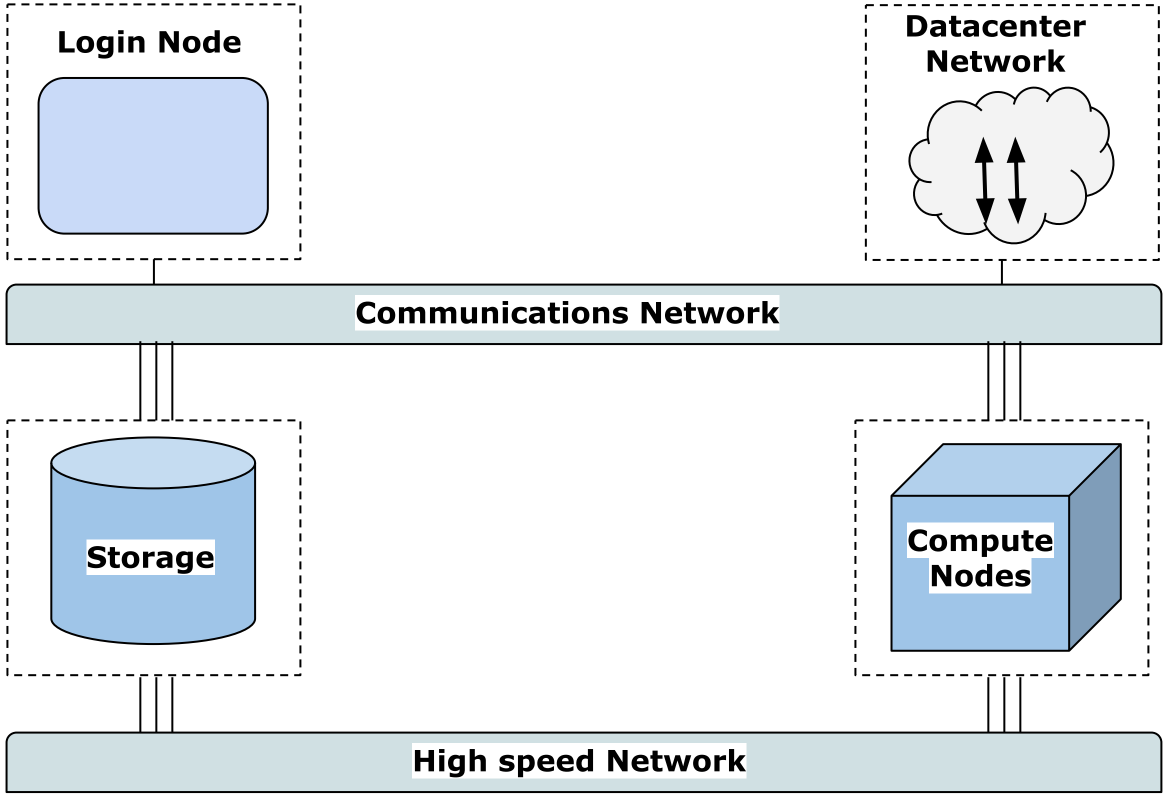
- User coordination required
- Resources can go unused
- Not scalable
- Difficult to maintain

Cluster

- Less user coordination required
- Efficient use of resources
- Highly scalable
- Standardized maintenance

- If you are utilizing dedicated nodes, you must consider factors such as user coordination of the resources, underutilization of the resources, and the inability to scale this approach.
- If you are utilizing a cluster environment, less user coordination is required, resources are efficiently utilized, and the approach is highly scalable.

Components of a Cluster



Hardware components will vary from cluster to cluster based on the need. Typical hardware you will find is:

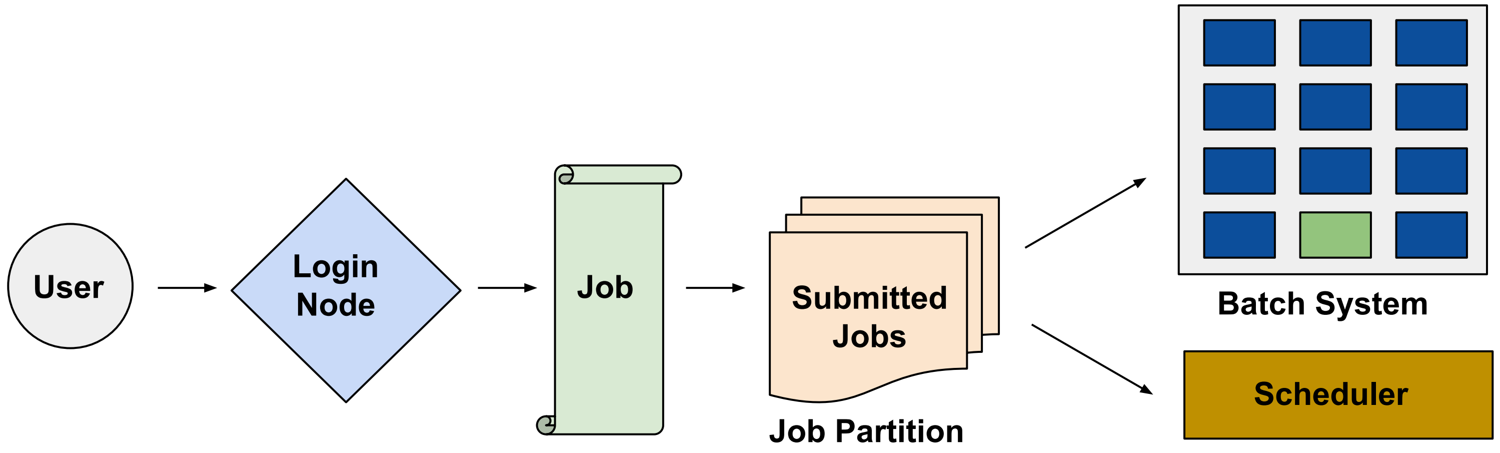
- Login Node
- At least 1 Network (communications and/or High speed)
- Storage (this may be on a separate server or could also be on the Login Node)
- A group of Compute nodes

Components of a Cluster

- **Login Node:** The login node is used for developing codes, preparing scripts to use with the scheduler, submitting and monitoring jobs in the scheduler, analyzing results, and transferring data.
- **Communications and/or High Speed Network:** The means of communication between all aspects of the cluster (nodes/storage/external networks).
- **Storage:** The location/locations that is shareable to the Login and Compute nodes in a cluster where data is stored.
- **Compute node**: A compute node is where the real work of the cluster is done. Compute nodes may contain specialized hardware to perform computations such as GPU, CPU, DPU, etc.
- **Scheduler:** Software that tracks resources available on the cluster while also scheduling submitted jobs in an organized manner.

What is a Scheduler?

- According to HPCWiki, a scheduler is software that implements a batch system on a cluster.
- Users do not typically run their jobs/calculations directly and interactively (as they do on their personal workstations or laptops), instead they submit non-interactive ***batch jobs*** to the scheduler.
- Although there is the ability to run interactive jobs when needed, the scheduler stores the jobs, evaluates their resource requirements and priorities, and distributes the jobs to suitable compute nodes



Scheduling basics. Scheduling Basics - HPC Wiki. (n.d.). Retrieved September 12, 2022, from https://hpc-wiki.info/hpc/Scheduling_Basics#:~:text=A%20scheduler%20is%20software%20that,batch%20jobs%20to%20the%20scheduler.

Slurm Scheduler

Slurm is an open source, fault-tolerant, and highly scalable cluster management and job scheduling system for large and small clusters running the Linux Operating System.

Slurm has three key functions:

1. Slurm allocates access to resources (compute nodes) to users for a duration of time.
2. Slurm also provides the ability for starting, executing, and monitoring jobs (either serial or parallel jobs) on nodes that are allocated.
3. Slurm also manages the queueing of pending work.

Slurm Workload manager. Slurm Workload Manager - Overview. (n.d.). Retrieved September 9, 2022, from https://slurm.schedmd.com/overview.html

Slurm Scheduler

Job submission parameters

`--ntasks=<x>`	Specifies how many instances are to be executed
`--nodes=<x>`	Specifies how many nodes are to be used
`--cpus-per-task=<x>`	Specifies how many cpu cores are to be used per ntask
`--partition=<partition name>`	Specifies which partition the job will be run on
`--time=HH:MM:SS`	Specifies the max time the job can run (Walltime)
`--gres=gpu:<x>`	Specifies to use GPU resources and how many GPUs to allocate
`--nodelist=<node name>`	Specifies running job on a particular node.
`--label`	Prepends the task number to line of stdout/err

Slurm Scheduler

Common Slurm commands

List all current jobs in the shared partition for a user:

```
squeue -u <username> -p shared
```

List detailed information for a job (useful for troubleshooting):

```
scontrol show jobid -dd <jobid>
```

List status info for a currently running job:

```
sstat --format=AveCPU,AvePages,AveRSS,AveVMSize,JobID -j <jobid>
--allsteps
```

Once your job has completed, you can get additional information that was not available during the run. This includes run time, memory used, etc.
To get statistics on completed jobs by jobID:

```
sacct -j <jobid> --format=JobID,JobName,MaxRSS,Elapsed
```

Slurm Scheduler

Common Slurm commands

List all current jobs for a user:

```
squeue -u <username>
```

List all running jobs for a user:

```
squeue -u <username> -t RUNNING
```

List all pending jobs for a user:

```
squeue -u <username> -t PENDING
```

List priority order of jobs for the current user (you) in a given partition:

```
showq-slurm -o -u -q <partition>
```

To view the same information for all jobs of a user:

```
sacct -u <username> --format=JobID,JobName,MaxRSS,Elapsed
```

Slurm Scheduler

Common Slurm commands

To cancel one job:

```
scancel <jobid>
```

To cancel all the jobs for a user:

```
scancel -u <username>
```

To cancel all the pending jobs for a user:

```
scancel -t PENDING -u <username>
```

To cancel one or more jobs by name:

```
scancel --name myJobName
```

To hold a particular job from being scheduled:

```
scontrol hold <jobid>
```

Slurm Scheduler

Common Slurm commands

To release a particular job to be scheduled:

```
scontrol release <jobid>
```

To requeue (cancel and rerun) a particular job:

```
scontrol requeue <jobid>
```

Slurm Scheduler

Slurm command examples

Interactive srun non-GPU

```
$ srun --ntasks=5 --nodes=1 --cpus-per-task=2 --partition=gpu --time=4:00:00 --label hostname
```

Output would look similar to:

```
0: dgx01
1: dgx01
2: dgx01
3: dgx01
4: dgx01
```

Slurm Scheduler

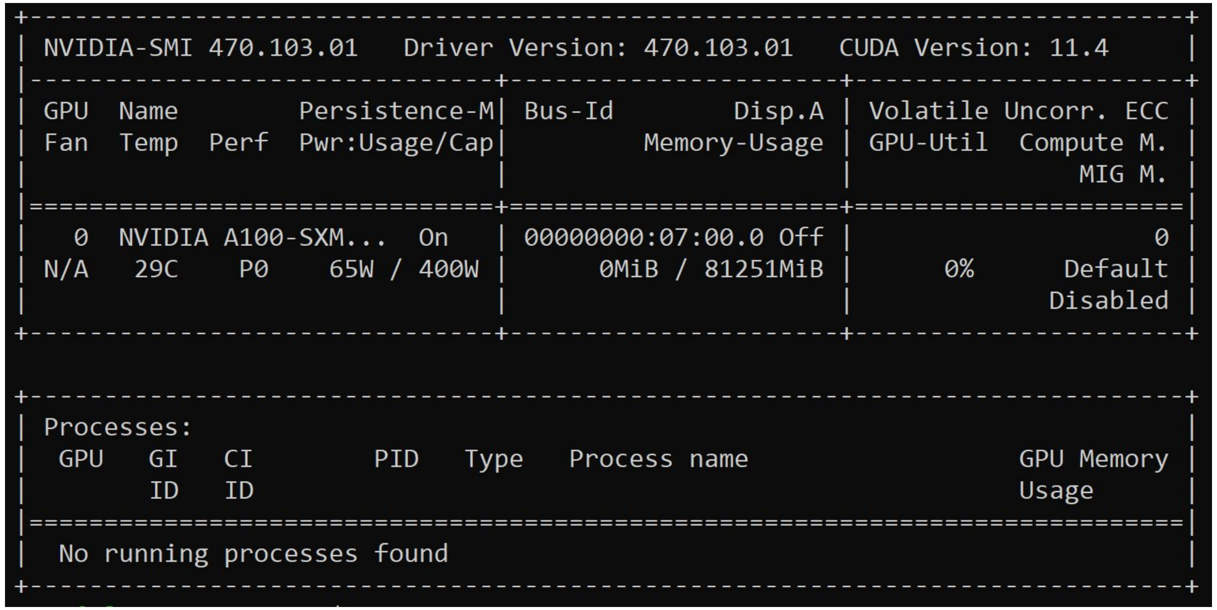
Slurm command examples

Interactive srun with GPU

```
$ srun --ntasks=1 --nodes=1 --cpus-per-task=1 --partition=gpu --gres=gpu:1 --time=4:00:00 nvidia-smi
```

This command allows 1 task utilizing 1 cpu and 1 gpu on the partition named ‘gpu’ with a wall time of 4 hrs and it will run nvidia-smi

Output would look similar to:



```
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 470.103.01   Driver Version: 470.103.01   CUDA Version: 11.4     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  NVIDIA A100-SXM...  On   | 00000000:07:00.0 Off |                    0 |
| N/A   29C    P0    65W / 400W |      0MiB / 81251MiB |      0%      Default |
|                               |                      |             Disabled |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

Slurm Scheduler

Slurm Batch Jobs

To run a Slurm job using sbatch, a job script needs to be created that instructs what application is going to be run and what resources are required.

The first line of the script needs to instruct the script is a bash script:

```
#!/bin/bash
```

Following that line will be `#SBATCH` entries that pass the options to SLURM

examples (there are more see : Slurm sbatch)

`#SBATCH -j <job_name>`	Names the job
`#SBATCH -p <partition>`	Instructs what partition job will run
`#SBATCH --nodes=<number>`	Requests the number of nodes allocated
`#SBATCH --ntasks=<number>`	Instructs number of tasks will be run
`#SBATCH --cpus-per-task=<number>`	Allocate number of cpus for each task

To run the sbatch job: `$ sbatch <job-script-name>`

Slurm Scheduler

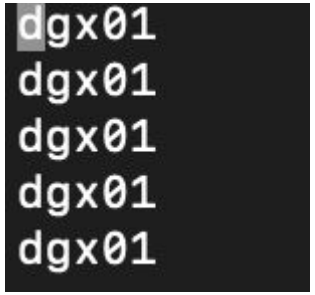
Slurm Batch Jobs

Batch mode non-GPU

```
#!/bin/bash
#SBATCH --partition=gpu     # Sets what slurm partition to use
#SBATCH --nodes=1                # Sets number of nodes
#SBATCH --ntasks=5               # Sets max number of tasks
#SBATCH --cpus-per-task=2        # Sets number of CPUs per task
#SBATCH --time=4:00:00           # Sets the walltime for the job

srun hostname
```

The output file slurm-<jobid>.out will show the hostname of the compute node the job ran on for the number of tasks. In this example is outputs the hostname 5 times in the out file.



```
dgx01
dgx01
dgx01
dgx01
dgx01
```

Slurm Scheduler

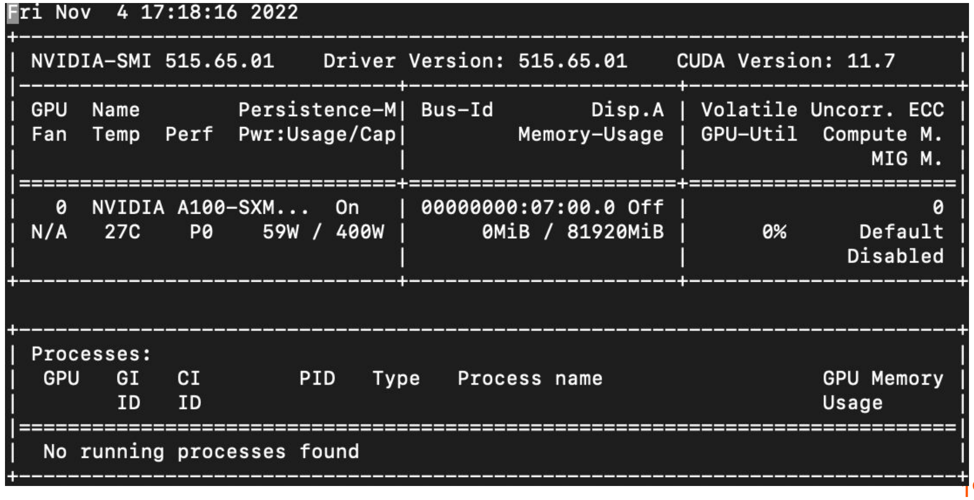
Slurm Batch Jobs

Batch mode with GPU

```
#!/bin/bash
#SBATCH --partition=gpu     # Sets what slurm partition to use
#SBATCH --nodes=1                # Sets number of nodes
#SBATCH --ntasks=1               # Max is event specific
#SBATCH --gres=gpu:1             # number of GPU’s to use
#SBATCH --cpus-per-task=2        # Sets number of CPUs per task
#SBATCH --time=4:00:00           # Sets the walltime for the job

srun nvidia-smi
```

Output in the job out file will show the nvidia-smi results.



```
Fri Nov  4 17:18:16 2022
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 515.65.01    Driver Version: 515.65.01    CUDA Version: 11.7     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  NVIDIA A100-SXM...  On   | 00000000:07:00.0 Off |                    0 |
| N/A   27C    P0    59W / 400W |      0MiB / 81920MiB |      0%      Default |
|                               |                      |             Disabled |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

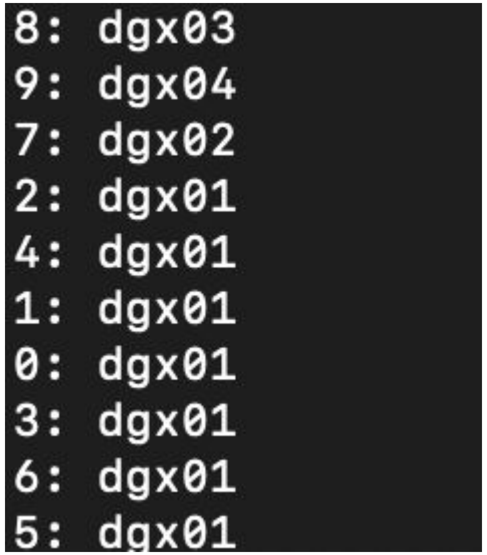
Slurm Scheduler

Slurm Batch Jobs

Basic Multi-node example

```
$ srun --ntasks=10 --nodes=4 --cpus-per-task=2 --partition=gpu --time=4:00:00 --label hostname
```

Output would look similar to:



```
8: dgx03
9: dgx04
7: dgx02
2: dgx01
4: dgx01
1: dgx01
0: dgx01
3: dgx01
6: dgx01
5: dgx01
```

OpenACC
More Science, Less Programming

Hands-On Interactive

- Check the status of the node and verify that the partition you are using is available

```
$ sinfo
```

- Check the status of running jobs

```
$ squeue $user
```

- Submit an interactive job (this job will give direct command line access to the compute node that gets assigned) that runs on 1 node, with 5 tasks and 2 cpus per tasks allocates 1 gpu for a time of 15 mins within the gpu partition.

```
$ srun --ntasks=5 --nodes=1 --cpus-per-task=2 --partition=<partition name>
--time=00:15:00 --gres=gpu:1 --pty /bin/bash
```

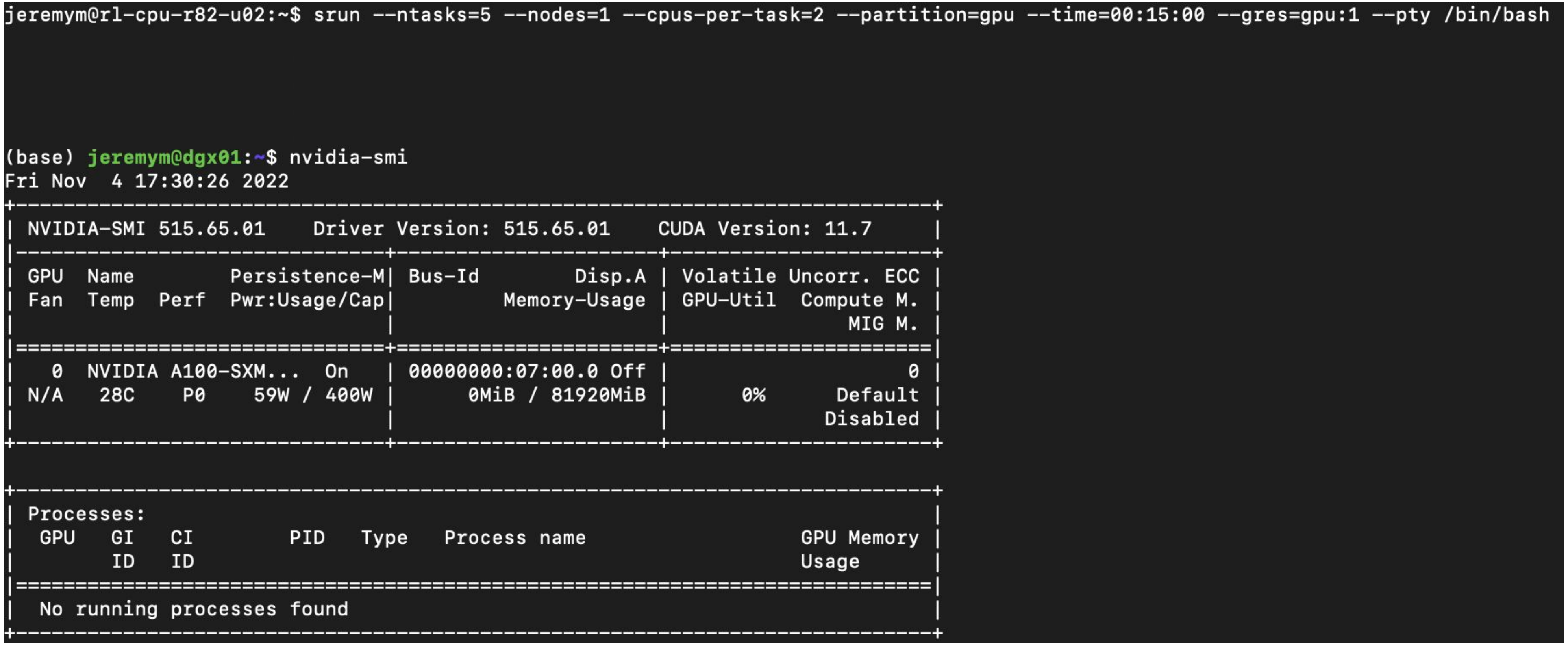
- Inside compute node verify GPU is visible

```
$ nvidia-smi
```

Result on next page

OpenACC
More Science, Less Programming

Hands-On Interactive



```
jeremym@rl-cpu-r82-u02:~$ srun --ntasks=5 --nodes=1 --cpus-per-task=2 --partition=gpu --time=00:15:00 --gres=gpu:1 --pty /bin/bash




(base) jeremym@dgx01:~$ nvidia-smi
Fri Nov  4 17:30:26 2022
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 515.65.01    Driver Version: 515.65.01    CUDA Version: 11.7     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  NVIDIA A100-SXM...  On   | 00000000:07:00.0 Off |                    0 |
| N/A   28C    P0    59W / 400W |      0MiB / 81920MiB |      0%      Default |
|                               |                      |             Disabled |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

Hands-On Non-interactive

- Check the status of the node and verify that the partition you are using is available

```
$ sinfo
```

- Check the status of running jobs

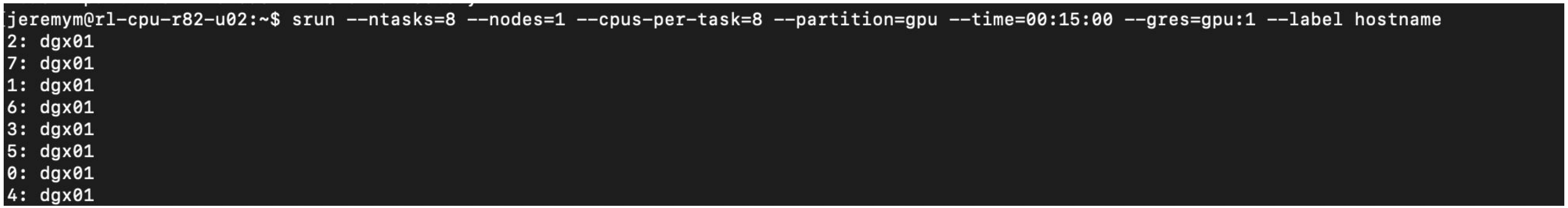
```
$ squeue $user
```

- Submit an non-interactive job (this type of job will run on the node without any interaction from the user) that will run on 1 node, with 8 tasks and 4 cpus per tasks and allocates 1 gpu for a time of 15 mins within a partition on the cluster.

```
$ srun --ntasks=8 --nodes=1 --cpus-per-task=8 --partition=<partition name> --time=00:15:00
--gres=gpu:1 --label hostname
```

Result on next page

Hands-On Non-interactive



```
jeremym@rl-cpu-r82-u02:~$ srun --ntasks=8 --nodes=1 --cpus-per-task=8 --partition=gpu --time=00:15:00 --gres=gpu:1 --label hostname
2: dgx01
7: dgx01
1: dgx01
6: dgx01
3: dgx01
5: dgx01
0: dgx01
4: dgx01
```

Resources and Links

- Additional resources
  - SchedMD
  - Open Hackathons technical resource page
  - Open Hackathons GitHub Repository

- Join the OpenACC and Hackathons Slack channel

- Licensing

Copyright © 2023 OpenACC-Standard.org. This material is released by OpenACC-Standard.org, in collaboration with NVIDIA Corporation, under the Creative Commons Attribution 4.0 International (CC BY 4.0). These materials include references to hardware and software developed by other entities; all applicable licensing and copyrights apply.

Learn more at

WWW.OPENACC.ORG

OpenACC
More Science, Less Programming